Übungsblatt 4

OPR2UE, Objektorientierte Programmierung, UE

Erreichbare Punkte	48 Punkte
Letzter Abgabetermin	Donnerstag, 8. Juni, 23:55 Uhr
Abgabedokumente	Quelldateien inkl. Tests
Abgabeort	Moodle (https://hagenberg.elearning.fh-ooe.at)

Aufgabe 1 - Prüfsummen 18 Punkte

a) Implementierung 12 Punkte

Prüfsummen erlauben es, Datenintegrität sicherzustellen. Eine der einfachsten Möglichkeiten, Prüfsummen zu bilden, sind Quersummen. Eure Aufgabe ist es, einige Algorithmen zur Ermittlung von Quersummen zu implementieren. Es sollen folgende Algorithmen umgesetzt werden:

- *Quersumme*: Bei der Quersumme werden die Ziffernwerte der einzelnen Bytes im String aufsummiert.
- *Einstellige Quersumme*: Die einstellige Quersumme wird im ersten Schritt genauso gebildet wie die zuvor beschriebene Quersumme. Ist das Resultat mehrstellig, wird von diesem wiederum die Quersumme gebildet. Dies wird so lange wiederholt, bis nur mehr eine Zahl übrigbleibt.
- *Alternierende Quersumme*: Bei der alternierenden Quersumme werden die einzelnen Ziffernwerte abwechselnd addiert und subtrahiert. (Dabei können auch negative Quersummen entstehen.)

Die Umsetzungen der Algorithmen müssen alle das Interface `Checksum` implementieren. Dieses Interface definiert die folgende Methode:

- `int checksum(String input);`

Beim Ermitteln der Prüfsummen der Texte arbeitet man nicht mit den String-Objekten selbst, sondern verwendet die dazugehörigen Byte-Arrays: `input.getBytes()`. Die Quersumme soll immer über die aneinandergereihten Bytes gebildet werden:

H	a	l	l	o		W	e	l	t	!
72	97	108	108	111	32	87	101	108	116	33

- Quersumme

  7+2+9+7+1+0+8+1+0+8+1+1+1+3+2+8+7+1+0+1+1+0+8+1+1+6+3+3 = 91

- Einstellige Quersumme

  7+2+9+7+1+0+8+1+0+8+1+1+1+3+2+8+7+1+0+1+1+0+8+1+1+6+3+3 = 91

  9 + 1 = 10

  1 + 0 = 1

- Alternierende Quersumme

  7-2+9-7+1-0+8-1+0-8+1-1+1-3+2-8+7-1+0-1+1-0+8-1+1-6+3-3 = 7

Achtet bei der Implementierung auf den richtigen Einsatz der Modifikatoren und vermeidet Codewiederholungen. Verwendet Vererbung und Interfaces, sofern möglich und sinnvoll.

b) Tests 6 Punkte

Testet die Implementierungen der Prüfsummen-Algorithmen mit JUnit-Tests.

Aufgabe 2 - Widerstände 30 Punkte

a) Basisimplementierung 16 Punkte

Widerstände sind elektrische Bauteile mit einem festen, unveränderbaren Widerstandswert (Einheit: Ohm, Ω). Aus solchen Widerständen können Schaltungen realisiert werden, deren Gesamtwiderstand wie folgt berechnet werden kann:

- Ein einzelner Widerstand ist eine Schaltung für sich. Der Gesamtwiderstand dieser Schaltung ist der Widerstandswert R des einzelnen Widerstands.
- Gesamtwiderstand einer Serienschaltung zweier Widerstände: $R = R_1 + R_2$
- Gesamtwiderstand einer Parallelschaltung zweiter Widerstände: $\frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2}$

Definiere zunächst ein Interface `Circuit` für Widerstandsschaltungen. Dieses Interface deklariert folgende Methoden:

- `double getOhm()`: Liefert den Gesamtwiderstand einer Schaltung.
- `int numberOfResistors()`: Liefert die Anzahl der Widerstände in einer Schaltung.

In weiterer Folge sollen Klassen für einen einzelnen Widerstand (`Resistor`), eine allgemeine zusammengesetzte Schaltung (`CompoundCircuit`) und für Reihen- (`Serial`) und Parallelschaltung (`Parallel`) entwickelt werden.

Die Klasse `Resistor` repräsentiert einen einfachen Widerstand mit unveränderbarem Widerstandswert. Die Klasse soll das Interface `Circuit` implementieren. Ein einfacher Widerstand kann keinen negativen Wert annehmen.

Die Klasse `CompoundCircuit` stellt eine allgemeine zusammengesetzte Schaltung aus zwei Objekten dar, die das `Circuit` Interface implementieren.

Darauf aufbauend sind die Klassen für Reihen- (`Serial`) und für die Parallelschaltung (`Parallel`) zu implementieren. Diese sind von `CompoundCircuit` abzuleiten. Beiden Klassen sollen Konstruktoren und die Methode zur Berechnung des Gesamtwiderstandes nach obigen Formeln zu Verfügung stellen.

b) Potentiometer 6 Punkte

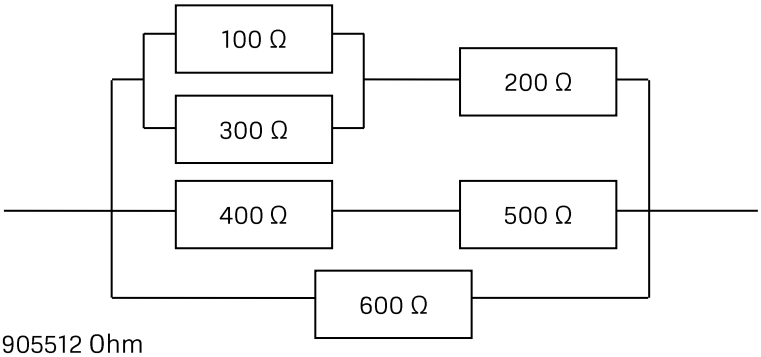
In dieser Aufgabe soll in das Klassensystem aus Aufgabe 1 eine weitere Klasse für ein Potentiometer hinzugefügt werden. Der Widerstand eines Potentiometers kann beliebig zwischen 0 und einem unveränderbaren Maximalwert eingestellt werden.

Die Klasse `Potentiometer` soll das Interface `Circuit` implementieren und des Weiteren eine Methode `setOhm(double ohm)` zum Setzen des aktuellen Widerstands innerhalb der Grenzen 0 bis zu einem unveränderbaren Maximalwert zur Verfügung stellen. Der Maximalwert soll über den Konstruktor einstellbar sein.

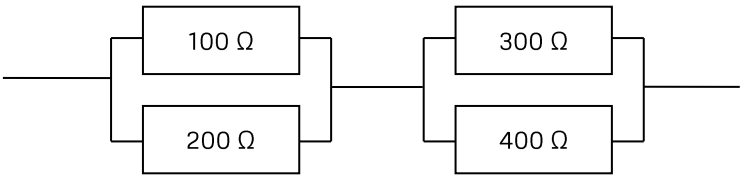
c) Tests 8 Punkte

Entwickle Testklassen, die die korrekte Implementierung der oben beschriebenen Klassen überprüfen.

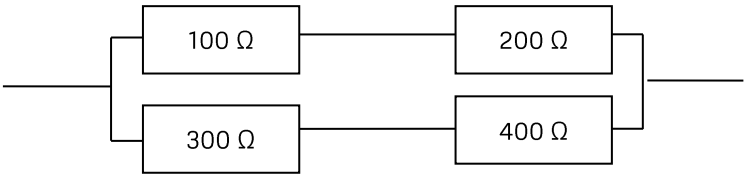
Beispielschaltungen:



Gesamt 155,905512 Ohm



Gesamt 238,095238 Ohm



Gesamt 210 Ohm